CS514: Intermediate Course in Computer Systems

Lecture 15: February 24, 2003

Overcoming network outages in applications that need high availability

Network outages

CS514

- We've seen a number of security mechanisms
  - VPNs
  - Firewalls and address translators
- And these run over the Internet itself
  - But problems can arise in the network
  - To what degree can an application "secure itself" against such outages?

Handling network failures

CS514

- Networks are themselves fault-tolerant
  - Assuming there is an alternative route!
  - Routing protocols detect problems and adapt when a destination becomes unreachable by its previously best route
  - But route changes don't happen instantly. Delays of many seconds or even minutes are common
- Moreover, network routing deals only with certain classes of failure
  - For example, *overload* is not treated as a good reason for quickly changing routing

A bit of history

CS514

- Prior to 1980, Internet routing was extremely responsive
  - But around 1980, studies showed that as much as 1/3 of the bandwidth of the Internet was consumed purely by routing protocols!
  - Moreover, the growth was worse than linear
- In early 1980's the concept of adaptive routine was revisited to address these problems

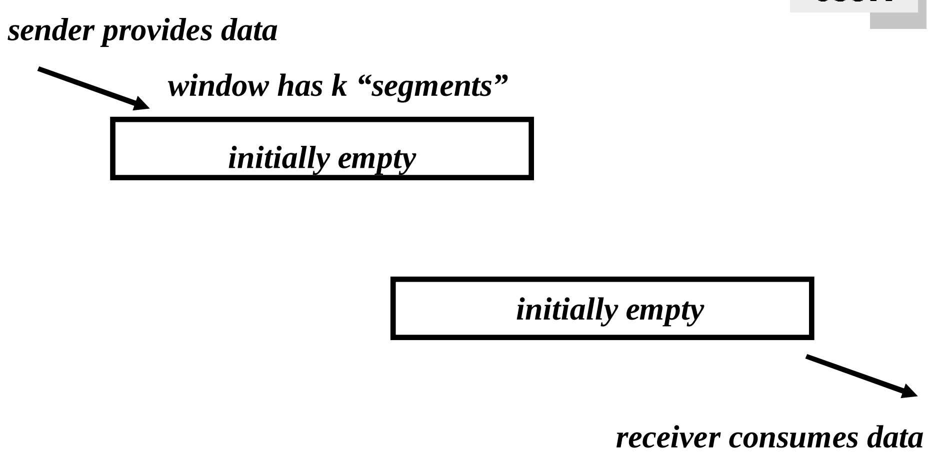
Streams concept revisited

CS514

- TCP supports "streams": reliable, point-to-point communication (also called a "flow")
- Like a telephone connection:
  - Information received in order sent, no loss or duplication
  - Call setup required before communication is possible (in contrast with basic message transport via UDP)
  - No message structure: abstraction is a stream of bytes
- Automatic flow control, error correction

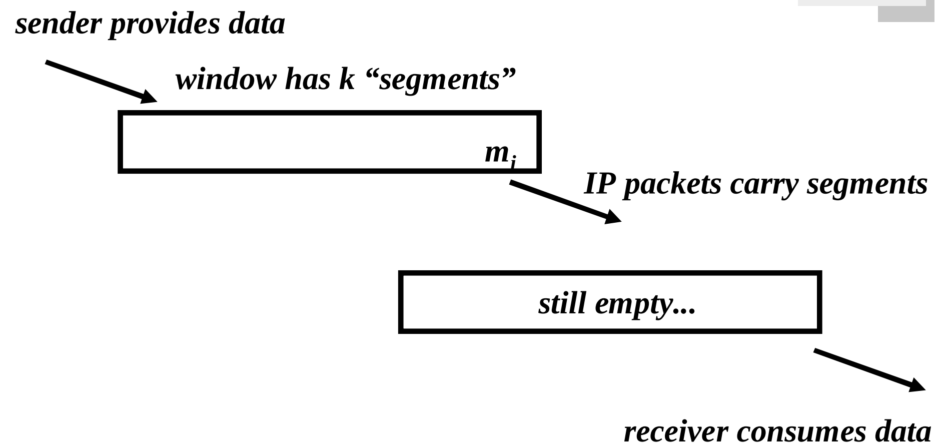
TCP sliding window

CS514



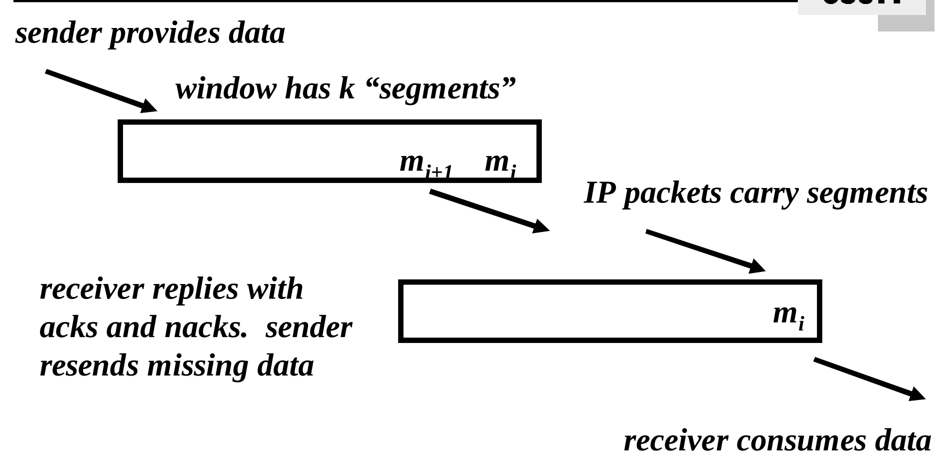
TCP sliding window

CS514



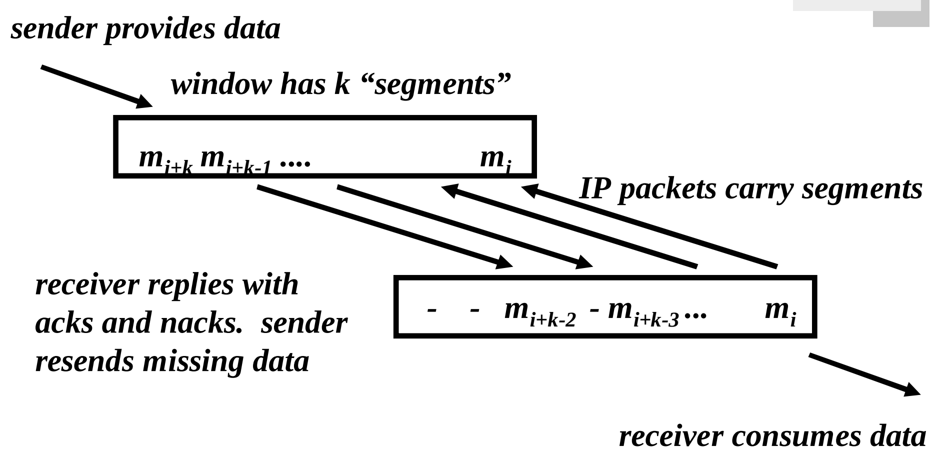
TCP sliding window

CS514



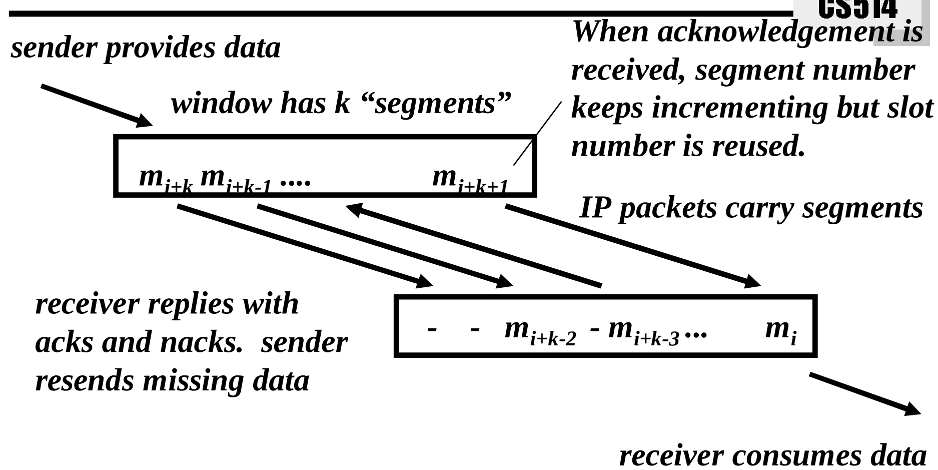
TCP sliding window

CS514



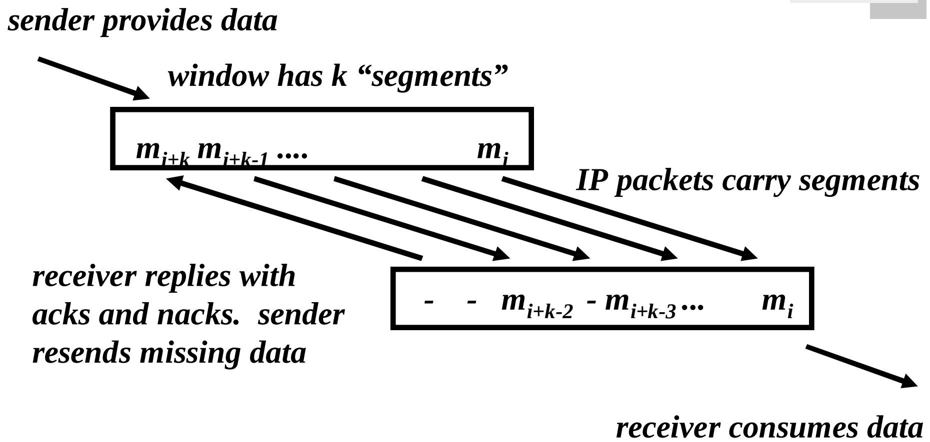
TCP sliding window

CS514



TCP sliding window

CS514



Why use a window?

CS514

- TCP wants to match
  - Rate that the sender is transmitting
  - Speed that the network can handle
  - Rate that the receiver can handle
- It wants to hide the end-to-end latency associated with the network
  - Applications want to do as few context switching or blocking events as possible
- So TCP acts like a bounded buffer with high and low-water marks, as used in the O/S

Typical implementation issues?

CS514

- When to send the ack
  - Send early: inefficient, channel clogged with acks
  - Send late: sender side fills window and waits
- When to send the nack
  - Send early: sender will send duplicates of all msgs
  - Send late: long delay waiting for desired data
- How big to make the window
- Send messages in “bursts”?

Where are the costs?

CS514

- Excess packets sent/received: very costly
  - Hence want minimal number of acks, nacks
  - Also want to avoid excess retransmissions
- Notice “tension” between sending acks/nacks too soon, and retransmission too soon, and between doing so too late.
  - Too soon: consumes bandwidth
  - Too late: leaves processes idle

Costs (cont)

CS514

- Delays on sender side:
  - Overheads associated with scheduling (e.g. if window fills up
- Avoiding “nervous” scheduling:
  - High-water/low-water mark scheme lets sender sit idle until there are several window slots free
  - Ideally, seek window size at which sender, receiver are rate matched and neither ever waits

Costs (cont)

CS514

- Delays on receiver side
  - Want a large enough window so that any error correction is “in the future” for receiver
  - Don’t want to delay nacks too long (else retransmission delayed too long)
- Nervous scheduling less of an issue here
  - Don’t use hi-water/low-water scheme in receiver

Timed approach

CS514

- Measure round-trip time (e.g. perhaps 1ms)
- Track rate of transmission for recent past
- Use to calibrate various constants:
  - Nack if a missing packet is late by 50% of expected time
  - Calibrate window to be 50-75% full in steady state
- Experience: very hard to make it work; variability in network load/latencies too big

Van Jacobson optimizations

CS514

- Dynamically adjust window size
  - While no loss detected, repeatedly increase size (linearly)
  - Detect loss: halve size ("multiplicative" backoff)
- Experience is very positive, many TCP's use this

Dealing with failures

CS514

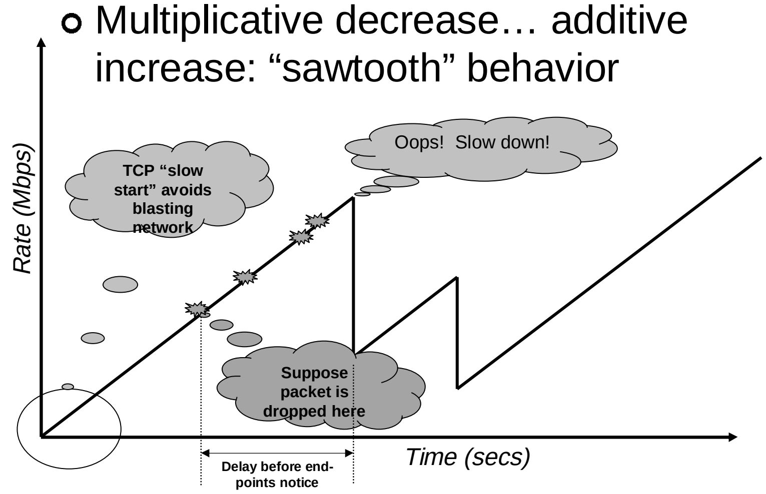
- Packets lost, duplicated, out of order: easy, just use sequence numbers (TCP calls these "segment" numbers)
- Sender or receiver fails, or line breaks:
  - After excessive retransmissions, or
  - After excessive wait for missing data, or
  - After not seeing "keepalives" for too long

  … ***break the connection and report "end of file"***

TCP rate at sender

CS514

- Multiplicative decrease… additive increase: "sawtooth" behavior



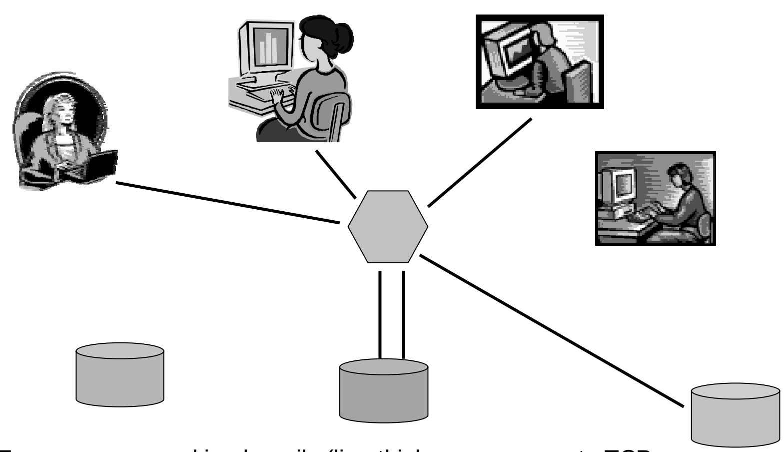
Issue of “catastrophic collapse”

CS514

- Suppose that many TCP flows run through the same router
  - They need time to notice packet loss and slow down
  - So they could overwhelm the router for many seconds before packet loss forces them to back off
- During this period the router basically dies
  - Worse still, a few aggressive flows could kill the router for all flows using it!

Sudden collapse

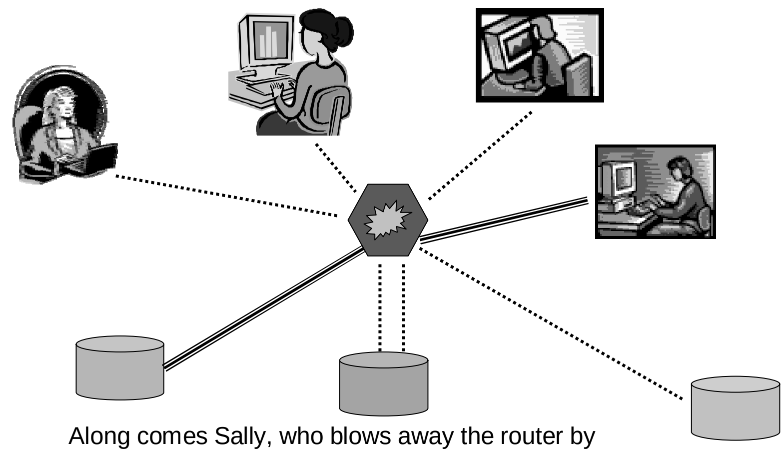
CS514



Everyone was working happily (line thickness represents TCP bandwidth)…. When suddenly….

Sudden collapse

CS514



Along comes Sally, who blows away the router by transferring a huge object at very high rates!

TCP slow start

CS514

- Try to avoid blasting router by having a TCP connection
  - Start sending slowly, not rapidly
  - Do this if it has been idle for more than a few seconds
- Idea is that we sort of “feel out the lay of the land” before overwhelming the network by accident

Random Early Detection (RED)

CS514

- Idea is to have the router anticipate that it is approaching a congestion state
  - Router notices this *well before* the problem actually arises
  - It signals the problem by randomly dropping some packets
- The faster a TCP flow is running the more likely it will lose some packets!
- So RED encourages fast TCP protocols to slow down before they mess up the router for everyone!

What does this add up to?

CS514

- End-user must expect
  - Variable TCP bandwidths
  - Potential loss of a TCP connection if an extended network outage occurs
  - Slow start after a period of idleness
- This behavior is as visible in a LAN shared with other users as it is in the Internet WAN

Back to routing…

CS514

- Recall that the Internet routes around faults
  - Clearly, if it routes around congestion, that will defeat the delicate dance between RED and TCP!
  - Moreover, back in 1980, *most* routing changes were triggered by load
- So the modern Internet is deliberately slow to reroute. Minutes, not seconds

Pulling it all together…

CS514

- So suppose you need to build a highly available application on a network
  - That network probably runs TCP on normal routers
  - So it will probably behave like the Internet if it gets heavily loaded!
  - Outages lasting minutes could be common…

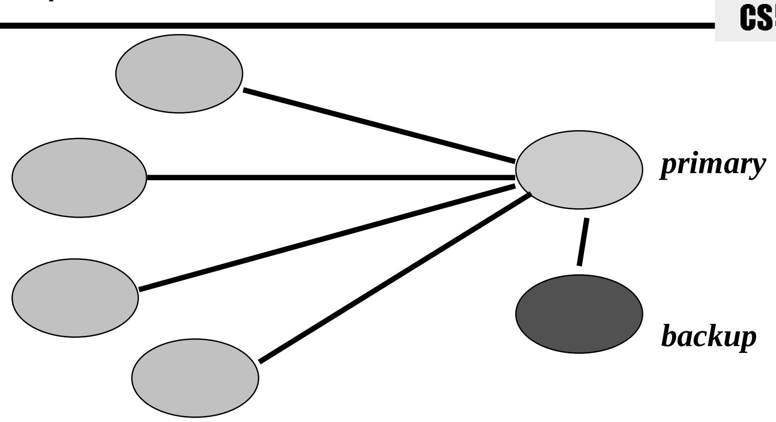
Other problems with TCP as a transport?

CS514

- A TCP channel can break because of a transient condition!
  - Example: overloaded machine, connection that temporarily fails, router crashes and must reboot itself (all are relatively common conditions)
- Systems with many TCP channels: *some may break but others stay connected!*
  - Famous example: FTP uses two connections between client and server. In fact one could break while other remains open!

Inconsistently broken TCP channels

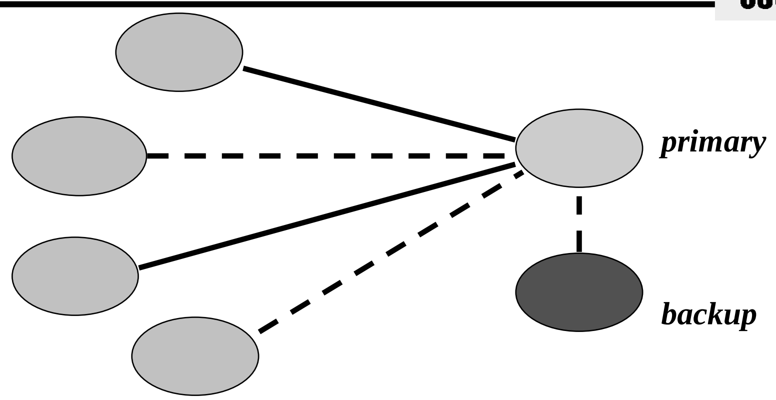
CS514



Clients initially connected to primary, which keeps backup up to date. (For example, in a database system)

Inconsistently broken TCP channels

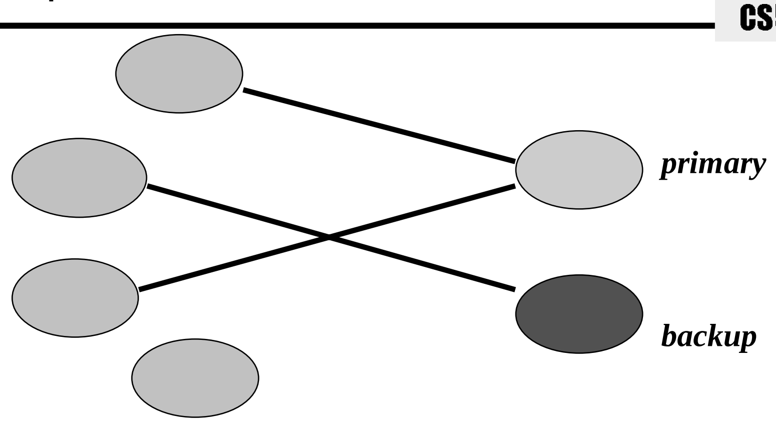
CS514



Transient problem causes some links to break but not all. Backup thinks it is now primary, primary thinks backup is down

Inconsistently broken TCP channels

CS514



Some clients still connected to primary, but one has switched to backup and one is completely disconnected from both

Why should this matter?

CS514

- Suppose that primary and backup are a service used for air traffic control
- Service tells controllers which parts of airspace are “available” for routing flights towards airport
- Primary and backup may try and give *different* controllers access to the *same* airspace! Each thinks it is “in charge” for the system as a whole!

Subtle semantics questions

CS514

- Are the “reliability semantics” of TCP actually different from those of RPC?
  - In both cases, what you “know” is limited to what has been explicitly acknowledged
  - Both can report “failures” when none has occured
  - Ultimately, TCP and RPC give same guarantees!
- Many systems run RPC over TCP as the “reliable” RPC option. Is this different from normal RPC?

Reliability/Consistency summary

CS514

- TCP connections can overcome loss of individual packets in communication layer
  - Remote method invocation protocols running on UDP also overcome such loss
  - Both forms of connections report failures inconsistently
- Not clear how either could be used to implement a “safe” primary-backup server for our ATC example!
- And application must struggle with unpredictable bandwidth…

Typical application-level issues

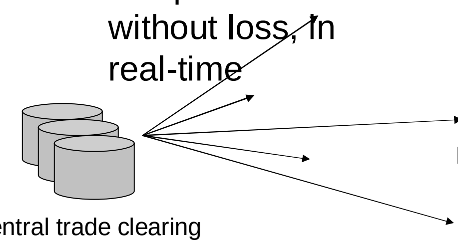
CS514

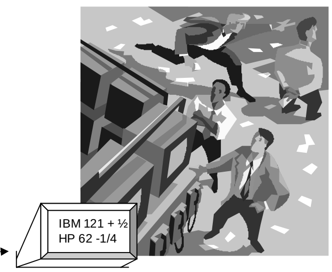
- Designers of the New York Stock Exchange system wanted to guarantee continuous availability
- But they also wanted to use standards as much as possible
- Could they “overcome” TCP’s limitations?
  - Aside: didn’t want to work with the new real-time versions of TCP, which are poorly supported (and may not work all that well even under the best circumstances).

Goal of the NYSE application

CS514

- Objective is to display selected data on overhead systems or other computers… without loss, in real-time





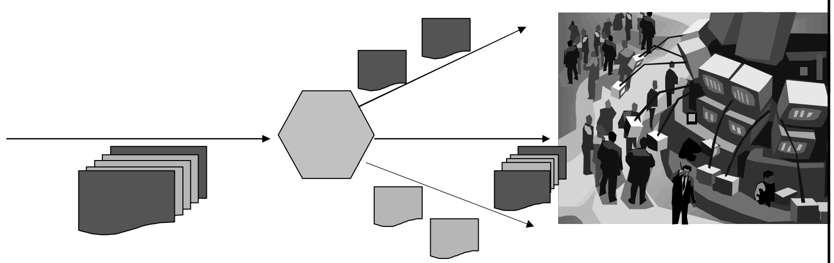
Built as a tree of application-specific routers

CS514

- Router knows what the data demands of "downstream" nodes are
  - E.g. on the right, {IBM,HP,SUN…}
  - On the left, {GTE,C,MOBL}
- Incoming message?
  - Check contents
  - Forward copies on each "leg" interested in message of this type

The NYSE as a tree of specialized routers

CS514



This is also called a "publish-subscribe" or a "content" routing mechanism….

Suppose we build this using standard technology?

CS514

- TCP connection could
  - Congest, leaving system "stuck"
  - Break even if router is healthy
  - Exhibit unpredictable behavior
- NYSE needed to build on TCP but prevent these undesired problems!

NYSE approach

CS514

- Begin by isolating the network
  - It has no outside traffic at all
  - Thus only load on a router is from the display subsystem itself
- This helps dampen the TCP congestion behavior

But transient failures and real failures are more of an issue

CS514

- A *transient failure* occurs if a machine hangs briefly
  - Common on both Linux or other UNIX systems and on Windows
  - With Linux, seen when a device driver prints error messages
  - With Windows, issue occurs when an application locks a critical kernel resource
- Such problem can cause a system to hang for 30 seconds or longer – long enough for TCP connections to start to break!

What about real failures?

CS514

- Issue here is that network can have many kinds of real failures
  - Routers can break
  - Cables can become lossy due to poor connections, noise, ground faults (usually due to moisture), etc
- Such problems may
  - Degrade a network but leave it limping
  - Cause bursts of failures
  - Cause real crashes
- They can be *very hard* to detect

Recall our goals

CS514

- We wanted continuous availability
  - Now we're faced with *many forms of single-point dependencies* and *many ways that a single failure could cripple the system!*
- What alternatives could be considered?

What about building directly on UDP?

CS514

- UDP would experience the same packet loss issues
  - RED also applies to UDP
  - And “real” failures impact UDP just as much as they impact TCP

NYSE opted for a third approach

CS514

- They decided to build a “dual network”
  - Duplicate the entire tree
  - Physically separate and duplicated hardware and power supply
- They assumed that transient problems or physical problems would be unlikely to disrupt both at once

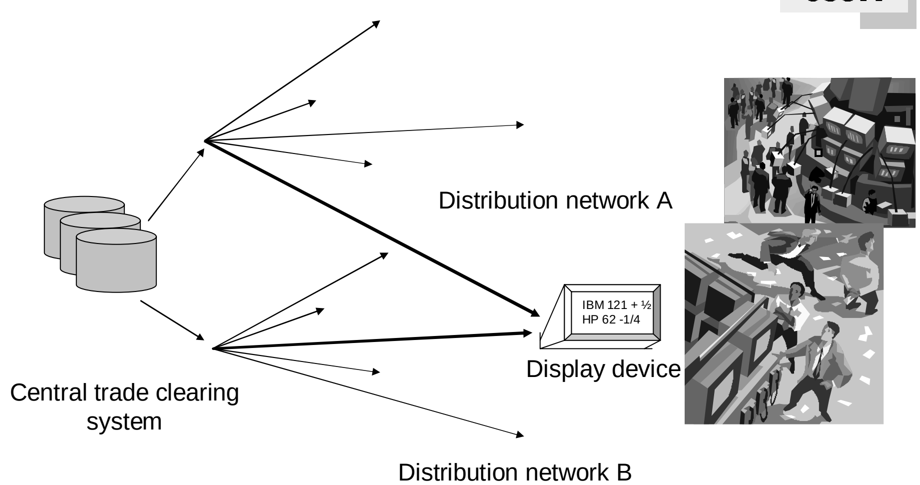
A new issue!

CS514

- Now they confronted difficulty of controlling network routing
  - Internet routing protocols could decide to route packets entirely on one network no matter what we “ask” them to do!
- Solved by giving each machine two IP addresses
  - One network handled one set of IP addresses (10.1.1.1, 10.1.1.2, etc) and the other the second set (10.2.1.1, 10.2.1.2…)

Replicating state remains a big challenge

CS514



Replication requirement?

CS514

- Two sets of routers with independent TCP links
  - But they need to behave in the identical way!
- This is another instance of the kind of replication issues seen before
  - Clusters need this kind of replication
  - So did our air traffic control system

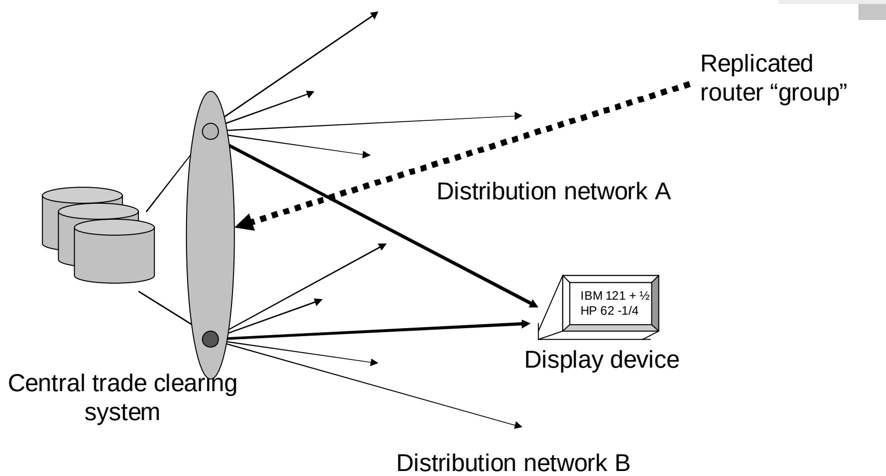
Forms of state (of data) that need to be replicated

CS514

- Set of data sources
  - If routers listen to different sets of sources, output will obvious be different
- Set of data receivers
  - If some router considers machine *x* to have failed and the other doesn't, *x* is at risk of single-point outages
- Data forwarding pattern
  - Routers should see identical inputs and forward data identically
- Even need to be consistent when changing the pattern!
  - Change forwarding pattern at some point in virtual time, not at an arbitrary moment
  - Use the sequence of messages as a time source

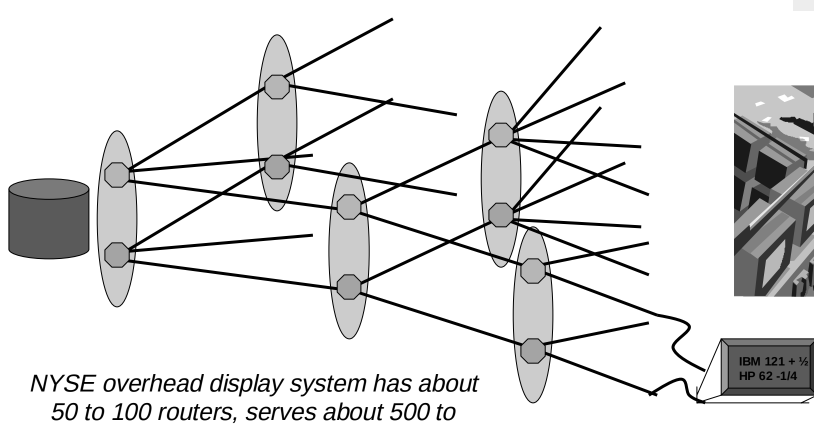
Replicating state remains a big challenge

CS514



Replicated router tree

CS514



NYSE overhead display system has about 50 to 100 routers, serves about 500 to 1000 display units, plus SEC, Dow, Reuters, Bloomberg, etc

NYSE used a package called Isis for this

CS514

- Ken wrote the system years ago
  - We'll discuss process group replication in more detail later, not important right now except as a useful mechanism
  - It includes a failure monitoring and reporting subsystem
- Isis itself needed to know about dual IP network
  - It was designed to use both networks concurrently as a way to ensure steady communication even during disruption

Additional real world issues?

CS514

- Many corporate networks would love to get the levels of availability seen in NYSE or similar systems
  - But technology like Isis is non-standard
  - Moreover, real networks face *aggressive threats*, not just passive ones

Denial of Service Attacks

CS514

- An increasingly common way to attack a network from the outside
  - A few recent events were insider attacks
- Goal of attacker is to flood routers or links and cause such high packet loss rates or loads that service is lost
- Often mounted from many attack sites
  - Hence *distributed* DOS or DDOS
  - Very hard to defend against

Challenge facing the most demanding, critical apps

CS514

- Either build a totally isolated network
- … or live in a world of
  - Transient outages
  - RED and greedy TCP behavior. TCP slow start
  - Long delays to detect and repair problems when failures or sustained overloads occur
  - IP routing that won’t make use of network redundancy even if present
  - DDOS attacks and other forms of intrusion… insider threats too!